REMARKS

This is a response to the non-final Office Action mailed on August 17, 2009.

The following claims are amended: 1, 4, 5, 7, 8, 11, 13, 28, 33, 41, 44, 45, 47, 52-54, 56-58, 62, 64, 65, 69, 70, 73, 77-83, 85-92, 97-100 and 102. The following claims are new: 103-113. No new matter is entered. Example support for the claims is as follows: claims 1, 33 (p.29, lines 18-19, p.31, lines 19-20, Fig. 9), claims 8, 62, 112 (p.12, line 19-p.13, line 2), claim 11 (p.24, lines 20-26), claims 56, 58, 64, 77-79 (incorporates claim 55, now cancelled), claim 65 (p.12, lines 9-13, incorporates claim 55, now cancelled), claims 103-105 (p.3, lines 17-19), claim 83 (p.13), claims 85, 86 (p.31, lines 4-11), claim 106 (Fig. 9, p.29, line 22-p.30, line 14), claim 107 (p.31, lines 14-18, Fig. 10), claim 108 (p.31, lines 25-27), claim 109 (p.31, lines 23-25), claim 110 (from previous version of claim 1), claim 111 (p.12, lines 22-25), and claim 113 (p.12, line 27-p.13, line 2). Other amendments for consistency and readability are also provided.

Rejection under Tuli, Davis and Dove

Claims 1, 13, 21, 22, 45, 47, 53, 54, 67, 69, 70, 83-88 and 102 are rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli (US 7,068,381) in view of Davis et al. (US 6,643,696), hereinafter referred to as Davis, and further in view of Dove et al. (US 2008/0034121 A1), hereinafter referred to as Dove. Applicants respectfully traverse the rejections.

Specifically, Applicant's claim 1 sets forth a specific process in which a markup language description of content is converted to script source code, and executable code is provided by compiling the script source code. The markup language description includes markup language source code which includes a tag name and associated attributes for a view instance element. This source code is used to create script source code, and the script source code is compiled. The script source code includes a script instruction which uses a function call. The script instruction includes the tag name and associated attributes in the function call. As an example, consider the markup language source code of a view instance (p.30, lines 1-15):

```
<window name="Contacts">
    <view x="5" y="10"/>
</window>
```

A parent tag name is `window`, as used in HTML. An attribute is: `name="Contacts"`.

This code can be converted to the following script source code:

```
LzInstantiateView(
      {tagName: "window", attrs: {name: "Contacts"}, children:
      [{tagName: "view", attrs: {x: 5, y: 10}}]}
)
```

The function LzInstantiateView is an example of an instantiate view function.

The script source code is compiled and provided to a renderer. At the renderer, an object is created and passed to an instantiate view function, e.g., to be viewed on a user interface (p.30, line 18-p.31, line 2).

The cited references taken alone or in combination simply do not disclose or suggest the claimed features. The Office should be careful to not impermissibly assemble the prior art in view of the cited invention.

Tuli sends bitmap or raster image data ("content") to a user device which is used to display an image, and the user uses a pointing device to click on the image. This event is sent to a server and processed to provide a "virtual click" to a "virtual browser" (col. 2, line 64 to col. 3, line 6).

Davis provides a process for tracking client interaction with a resource such as a web page downloaded from a server. In particular, an HTML document is downloaded from a server (e.g., server A, Fig. 3), then images specified by first embedded URLs are downloaded. A second embedded URL in the document points to a first executable program that runs on a server (e.g., server B). A third embedded URL in the document points to a second executable program that is downloaded to and runs on the client. When the images specified by the first embedded URLs are downloaded, the first executable program on the server runs. The server can capture identifying information from the client. The second executable program determines the current time when it initializes, and the current time when the user leaves the HTML document. The elapsed time is then reported to the server (col. 5, line 40-col. 6, line 16).

Dove provides a graphical program from a graphically based programming environment, such as where the user uses a block diagram editor. The graphical program may be represented as data structures that define or specify the operation of a graphical program. The data structures can be converted to an executable format such as machine language code or an interpretable script or other similar executable format, for execution on a portable device (par. 10, 22). Dove seeks to avoid high-level text-based programming languages which are translated to the machine language

level by translators known as compilers or interpreters (par. 3).

While Dove provides an interpretable script, Dove does not compile the script to executable code. Thus, the cited combination, even if made, *arguendo,* still does not result in the claimed invention. Moreover, *the cited references do not provide the specific conversion of markup language source code to script source code, and compiling of the script source code to provide executable code as claimed. The cited references do not disclose or suggest how tags and associated attributes of markup language source code can be provide in script source code, in a manner which allows rendering and interaction on a user interface.*

Regarding Harrington, which is cited below for using ActionScript in a Flash player in a browser, there is still no teaching of the specific conversion and compiling as claimed.

Claim 1 and its dependent claims, and the related claims, are therefore clearly patentable.

For example, claim 13 sets forth that a first application runs on a user device, and provides a request to a server for a second application. The second application is accessed, compiled and transmitted to the user device. Regarding Tuli at col. 2, lines 5-13, this refers to a web server receiving a web page, electronic message, HTML or JAVA information. However, such information is not an application. Moreover, Davis at col. 5, lines 54-58 refers to a second executable program that is downloaded to, and runs on, a client. However, it does not call another application as claimed. Claim 13 is therefore clearly patentable.

The Examiner is requested to clarify the rejection of claim 45 as the Office Action refers to a mark up language description which is not in the claim, and does not refer to specific features which are in the claim.

Rejection under Tuli, Davis, Dove and Rubin

Claims 4, 7, 52 and 73 are rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli, Davis and Dove in view of Rubin et al. (US 6,701,522), hereinafter referred to as Rubin. Rubin states that plug-ins are auxiliary programs added to web browsers that provide them with new functionality (col. 7, lines 21-23).

Regarding claim 4, the Office Action asserts that one of ordinary skill would have been motivated to combine Rubin with Tuli because plug-ins are auxiliary programs added to web browsers that provide them with new functionality (Rubin, col. 7, lines 21-23). However, Tuli specifically seeks to avoid the need for a mini-browser which requires a powerful microprocessor

(col. 1, line 66 to col. 2, line 4) and does this by moving the browser functionality to the server as a virtual browser (col. 2, lines 9-24). Accordingly, a person of ordinary skill in the art would see that the user device of Tuli does not have a browser functionality which can support a plug-in program. Claim 4 is therefore clearly patentable.

Regarding claim 52, Dove at par. 22 indicates that data structures of a graphical program can be converted to an executable format, such as machine language code or an interpretable script. Tuli is cited as provided a request to a web server. However, there is no disclosure or suggestion of providing executable code in response to an indication received at a server from a user device.

Regarding claim 73, while a Flash player *per se* is known, the cited references do not provide executable code that can be used by a Flash player. Applicants provide a specific technique for creating such executable code that is neither disclosed nor suggested by the cited references.

Rejection under Tuli, Davis, Dove and Rubin

Claims 5, 8, 51, 81, 89-97 and 101 are rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli, Davis and Dove in view of Harrington (US 2002/0156909).

Regarding claim 5, Harrington is cited for using ActionScript in a Flash player in a browser. However, Applicants are not merely using ActionScript in a Flash player. Applicants provide a way to convert a markup language description of content to script source code, and to provide executable code by compiling the script source code as ActionScript source code into corresponding byte code in the executable code. The cited art fails to disclose or suggest this feature.

Claim 8 sets forth that markup language source code includes a source attribute which references a name of a media file, and the source attribute is within a window tag in the markup language source code. Further, the media file is transmitted with the executable code from the server to the user device for use by a rendering entity at the user device in allowing the user to access the video in a window on the user interface when the media file is referenced by the source attribute within the window tag when the executable code is executed. The cited art fails to disclose or suggest this feature.

Claim 81 sets forth providing an object in the executable code which includes fields storing attributes which identify the name and a format of a media file. Tuli at col. 4, lines 18-22 refers to using JPEG to compress a color image which is provided to the remote user device. However, there is no mention that an object which includes fields storing attributes which identify the name and a

format is provided via a user interface.

Regarding the transcoding of claims 89-92, and also regarding claims 92-97 and 101, while such transcoding has been used *per se*, Applicants provide transcoding in a specific context. Regarding Tuli's dividing an image into sections after a bitmap or raster is created, this does not involve transcoding in the sense which would be understood by those skilled in the art, nor does it involve transcoding between specific formats as claimed. Moreover, Harrington provides no mention of transcoding. *Regarding the suggestion that Tuli could perform transcoding as claimed, Tuli teaches against this suggestion as Tuli is concerned with limiting his device to render a bitmap image of a web page. A person of ordinary skill would not seek to modify Tuli's device to handle media content such as audio and video (or transcoding of such content), which are incompatible with Tuli's stated purpose.*

Rejection under Tuli, Harrington and Rubin

Claims 28, 30, 31, 62 and 80 are rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli in view of Harrington and Rubin. Claim 28 sets forth, in part, that a markup language description uses a source attribute to reference the name of a media file comprising at least one of audio, video and a movie. Further, executable code is provided based on the markup language description, and at a plug-in, content is rendered based on the executable code, the source attribute and the media file. An example of a source (src) attribute is `src="image.swf"` at p.12, line 27-p.13, line 2 of the specification. First, as mentioned, Tuli cannot handle, and teaches against, a rendering device for audio, video and a movie. Further, Tuli does not use any source attribute which references a media file when executing code.

Regarding claim 62, while .SWF files are known *per se*, Applicants provide a specific technique in which a markup language description is compiled to provide executable code, and .SWF files are referenced in the markup language description by a tag within a window tag, so that the .SWF file is played within a window on a user interface when a plug-in renders content. Even if, *arguendo*, Tuli's device could play .SWF files, there is still no disclose or suggestion of Applicants' specific technique for playing an .SWF file in a window, using executable code obtained from a markup language description which reference the .SWF files in the manner claimed.

Rejection under Tuli, Davis, Dove and Russell

Claim 11 is rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli, Davis and

Dove in view of Russell (2002/0069420). Claim 11 sets forth that different types of authenticating are provided for different types of content, including content types of application, data and service. Russell is concerned with delivering content such as movie files over a network. Russell states that a network may authenticate a request to download a content item (such as a movie) by verifying that: (a) the user has purchased a license for the requested content item, (b) the user is in a geographical region that is allowed to download the requested content item, and (c) the user is not attempting to download the content item more times than is reasonable (par. 94). However, there is no disclosure or suggestion of using different types of authenticating for different types of content as claimed.

Rejection under Tuli, Harrington and Rubin

Claims 33, 41-44, 55-58, 60-62, 64, 65, 77, 78 and 98-100 are rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli in view of Harrington and Rubin.

Claim 33 and 41 are patentable for similar reasons as discussed in connection with claim 1.

Claim 44 is patentable for similar reasons as discussed in connection with claim 28.

Claim 58 is patentable for similar reasons as discussed in connection with claim 8.

Applicants believe that claim 62 was inadvertently included under this rejection as claim 62 was only addressed at par. 42 of the office action, in the rejection under Tuli, Harrington and Rubin.

Claim 65 sets forth that an element of a markup language description is identified by a markup language tag, and provides an inline definition of vector graphics using an svg tag and associated attributes within a window tag, in response to which the vector graphics are rendered in a window on a user interface when executable code is executed. Tuli at col. 2, lines 59-63 or elsewhere provides no disclose of any such technique.

Rejection under Tuli, Harrington, Rubin, Dove and Davis

Claims 68 and 79 is rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli, Harrington, Rubin and Dove, in view of Davis. Claim 68 is cancelled. These claims are patentable at least by virtue of their dependence on independent claims, which are patentable for reasons mentioned previously.

Rejection under Tuli, Davis, Dove, Harrington and Rubin

Claim 82 is rejected under 35 U.S.C. §103(a) as being unpatentable over Tuli, Davis, Dove, Harrington and further in view of Rubin. This claim is patentable at least by virtue of its dependence on independent claim 1, which is patentable for reasons mentioned previously.

New claims 103-112

Claims 103 and 105 recite specific objects that are created by an instantiation view function and displayed on a user interface. The cited references do not disclose or suggest these features as they do not provide an instantiation view function as claimed, where the instantiation view function is called by a script instruction in script source code as claimed. Moreover, Tuli, for instance cannot create a sound object since only bitmap data is rendered.

Claim 106 sets forth that a markup language description includes a parent tag name and associated attributes, and a related child tag name and associated attributes. An example is provided by:

```
<window name="Contacts">
     <view x="5" y="10"/>
</window>
```

A parent tag name could be `window` while a child tag name is `view`. Further, a script instruction includes both the parent and child tag names and their attributes. Objects displayed on a user interface are created based on the parent and child tag names and their attributes. The cited art provide no such details as they are not concerned with converting script source code to executable code as claimed. Claim 107 is similar patentable as no specific technique for calling an instantiation view function as claimed is provided by the cited art.

Similarly, regarding claims 108 and 109, the cited art provide no mention of the use of a table of instantiation functions as claimed.

Claim 110 sets forth that the markup language description uses the source attribute (which references the name of the media file) within an image tag, and the media file comprises an image file. Claim 111 sets forth that the markup language description uses the source attribute within an image tag, and the media file comprises an image file. Claim 112 sets forth that the markup language description uses the source attribute within a window tag, in response to which the plug-in renders the media file in a window on the user interface.

An example at p.12-13 of the specification is:

```
<window>
  <image src="image.jpeg"/>
  <button>Okay</button>
     </window>
```

The cited art simply provides no disclosure or suggestion of these features.

Claim 113 sets forth that the markup language description uses the source attribute within an swf tag, and the media file is a .SWF file. See the discussion above regarding claim 62

Conclusion

In view of the above, each of the pending claims is believed to be in condition for immediate allowance. The Examiner is therefore respectfully requested to pass this application on to an early issue.

The Examiner's prompt attention to this matter is greatly appreciated. Should further questions remain, the Examiner is invited to contact the undersigned attorney by telephone.

The Commissioner is authorized to charge any underpayment or credit any overpayment to Deposit Account No. 501826 for any matter in connection with this response, including any fee for extension of time, which may be required.

Respectfully submitted,

Date: November 17, 2009

By: /Ralph F. Hoppin/
Ralph F. Hoppin
Reg. No. 38,494

VIERRA MAGEN MARCUS & DENIRO LLP
575 Market Street, Suite 2500
San Francisco, California 94105-4206
Telephone: (415) 369-9660, x214
Facsimile: (415) 369-9665